01UDFOV/01TXYOV Applicazioni Web I / Web Applications I [2022/2023]

Lab 11: Multi-user Application with Authentication

In this lab, you will do two exercises to complete the FilmLibrary app: a) to commit the work developed so far on GitHub Classroom – to experiment with the platform in preparation of the exam – and b) to implement the login. After completing the entire lab, you are free to push again on your GitHub Classroom's private repository.

1. Commit your work to GitHub Classroom

As reported in the exam rules, you will use GitHub Classroom to submit your web application for the evaluation. To avoid last-minute issues, this exercise is to verify that you are successfully able to use the GitHub Classroom platform.

According to the course you belong to, you can submit what you developed so far at the following links:

- Applicazioni Web I: https://classroom.github.com/a/g5Hi-YAN
- Web Applications I [A-H]: https://classroom.github.com/a/m6R4j53E
- Web Applications I [I-Z]: https://classroom.github.com/a/1Xi3Mgf6

You can find instructions on how to use and submit on GitHub Classroom in the linked PDF document.

NOTE: the submitted application will not be NOT evaluated in any way. It is just a way for you to become familiar with the system.

2. Login and Logout Your Users

You will add the possibility of **having multiple users** in your application, enabling them to **authenticate** (i.e., *login* and *logout* functionalities) for managing their films. Non-authenticated users won't see a list of films anymore.

- In the front-end, add a new page (with a dedicated route) with a form, which will be used to log in. The page should be well structured in terms of needed components and appropriate states. The login form will have two *mandatory* fields: **email** and **password**. The login form should be validated before its submission, and you must use proper error messages when inconsistencies are found. Specifically, at least the following checks should be executed:
  - When a field is missing or empty it must be forbidden to send a log-in request to the server.
  - The email should be properly formatted (i.e., *something@something.something*).
- In Express, implement the **login** process by exploiting the **Passport** authentication middleware.
  - Add a new user with username "*testuser@polito.it*" and password: "*password*" in the database.
    Beware: do not store **plain text passwords** in the database! Use **scrypt** (*see the hints*) to generate a hash of the passwords before saving them.
  - Create at least three new films in the database and assign them to the newly created user.
- When the login process **fails**, the front-end should display a suitable error message (e.g., "*Incorrect username or password*") and continue to show the login form. Instead, when the login is **successful**,

the application redirects the users to their film list page, showing a message like: "*Welcome, {name_of_the_user}*".

- Identify the routes that need to be authenticated (e.g., those to get or modify the list of films) and *protect* them accordingly. Non-authenticated users must not see any film, meanwhile authenticated users must see **only** their films.
- Implement the **logout** functionality, again by exploiting the **Passport** authentication middleware. When the users are logged out, redirect them to the login form.

Finally, if you want, push the solution of this lab on the private repository created for you by GitHub Classroom.

Hints:

1. *Note: the password of the users already inserted into the database is* `password`.
2. *You can use the following website to generate the hash of a password, according to scrypt: https://www.browserling.com/tools/scrypt.*
   If you need to generate a salt by hand, you can use https://www.browserling.com/tools/random-hex, considering the length of the generated hex as the one of the salt (e.g., 16).